TRANSIT: Specifying Protocols with Concolic Snippets

Abhishek Udupa, Arun Raghavan,
Jyotirmoy V. Deshmukh¹, Sela Mador-Haim,
Milo M. K. Martin and Rajeev Alur

University of Pennsylvania
¹Currently with Toyota Technical Center
Distributed Protocols

- Distributed Protocols:
  - Multiple communicating agents
  - Asynchronous and unordered communication
  - Global correctness requirements

- Challenging to get right

- Success story for verification
  - But still must be specified in the modeling language
The Traditional Specification Methodology
Traditional Specifications

Transitions from I to S.

1. GetS
   - Req (I→S)
   - Dir (I→S)
   - S→S

2. Data

3. Data

1. GetS
   - Req (I→S)

2. Fwd-GetS
   - Dir (I→S)
   - Owner (M→S)

3. Data

1. GetM
   - Req (I→M)
   - Dir (I→M)
   - (2) Data[ack=0]

2. Fwd-GetM
   - Req (I→M)

3. Data[ack=0]

1. GetM
   - Req (I→M)

2. Fwd-GetM
   - Dir (M→M)

3. Owner (M→I)
Traditional Specifications

Rule

\[
\text{State} = \text{D\_BUSY} \land \text{InMsg.MType} = \text{UNBLOCK\_S} \implies
\]

Begin

\[
\text{State} = \text{D\_M};
\]

\[
\text{sharers} = \text{SetUnion} (\text{Sharers}, \text{SetOf(InMsg.Sender)});
\]

\[
\text{SendMsg} \{ \text{Type} = \text{ACK}, \text{Acks} = 1, \text{InMsg.Sender} \};
\]

EndRule;
Traditional Specifications

- Invariants
- Model Check/Verify
- Verified Protocol
- Counter-example
Traditional Specifications

I know exactly what to do in this particular scenario!!

BUT...

Still need to code it up

Model Check/Verify

Verified Protocol

Invariants

Counter-example
Can we make the process of specifying distributed protocols easier?
Our Approach to Specification

Scenarios:
- Describe execution traces
- Translated from informal specs
- Can be symbolic
- Can be concrete
Our Approach to Specification

<table>
<thead>
<tr>
<th>Transaction</th>
<th>load</th>
<th>store</th>
<th>replacement</th>
<th>FWkGetS</th>
<th>FWkGetM</th>
<th>Inv</th>
<th>Put-Ack</th>
<th>Data from Dir(old)</th>
<th>Data from Dir(new)</th>
<th>Data from Owner</th>
<th>Inv-Ack</th>
<th>Last-Inv-Ack</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IS^D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IM^AD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IM^A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM^AD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SM^A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MI^A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SI^A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I^A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Our Approach to Specification

Protocol Skeleton:

- Communication architecture
- Message types
- Set of processes
- State variables for processes
Our Approach to Specification

I know exactly what to do in this particular scenario!!

AND...

That’s EXACTLY what I’m going to specify with additional snippets!!
Goal

Protocol Skeleton

+ Scenarios

+ Invariants

下去

Completed Protocol Specification
Snippets and Scenarios

- Scenarios:
  - Sequence of message exchanges/transitions
  - Transcribed from informal specs
  - A collection of *snippets*

- Snippets:
  - Describe actions on a *single* transition
  - Relate current values of variables to updated values
Symbolic Snippets

Cache 1
State = S
State = S_M
Pend = Msg.Acks;

Directory
WR_REQ
RSP { DATA, ACKS }
INV { REQ = 1 }
ACK { CNT = 1 }

Cache 2

Pend -= Msg.Cnt;
Pend = 0 ➔
  State = M;
Pend != 0 ➔
  State = S_M;
Concrete Snippets

Cache 1

State = S
State = S_M
Pend = 1;

Directory

WR_REQ
RSP { DATA, ACKS = 1 }
INV { REQ = 1 }

Cache 2

Pend = 0;
State = S_M;

ACK { CNT = 1 }
Summing up Snippets

- Snippets are Boolean-valued formulas
- Suppose we need x to be max(a, b)
- Could specify it using a few concrete snippets:
  - $a = 5 \land b = 10 \implies x = 10$
  - $a = 8 \land b = 5 \implies x = 8$
  - $a = 0 \land b = 2 \implies x = 2$
- Or using symbolic snippets
  - As a constraint: $(x \geq a \land x \geq b \land (x = a \lor x = b))$
- Or as the desired code itself:
  - $x := a > b ? a : b$

Snippets more flexible than code
Solving for Snippets

To generate a completed protocol TRANSIT needs:

Find expressions consistent with given snippets

- Rule
- Expression grammar: Int, Bool, BitVector types
- Arithmetic, bit-vector and conditional operations
Expression Inference

• Consider only concrete snippets for the moment

• Enumerated in increasing order of size

• Consistency check involves a simple evaluation

Diagram:

1. Get Next Expression
2. Consistent with all Concrete snippets?
   - Yes: Output e
   - No: Repeat

19
Expression Inference: Pruning

• We’re still assuming all concrete snippets
  \{ a : 3, b : 4 \}
  \{ a : 5, b : 3 \}

\begin{align*}
\text{indist} \\
\text{indist}
\end{align*}

• Expressions are built bottom-up

• All expressions are retained
Expression Inference: Pruning

• We’re still assuming all concrete snippets
  \{ a : 3, b : 4 \} \quad \{ a : 5, b : 3 \}

- Pruning an expression of size $k$:
  - Reduces expressions of size $> k$ that are considered
Expression Inference - CEGIS

• Now consider symbolic snippets
• Satisfying assignment to symbolic snippet = Concrete Snippet!
• Use Counter-example guided Inductive Synthesis (CEGIS)

```
Get Next Expression

SolveConcrete

Consistent with all Concrete snippets?

Yes: Output e
No
```

22
Expression Inference - CEGIS

• Now consider symbolic snippets
• Satisfying assignment to symbolic snippet = Concrete Snippet!
• Use Counter-example guided Inductive Synthesis (CEGIS)

Diagram:
- SolveConcrete
- Concrete Snippets
- Consistent with all Symbolic snippets?
  - Yes: Output e
  - No: Add Concrete Snippet corresponding to the witness for inconsistency of e
Evaluation
Cache Coherence Protocols

- **TRANSIT** for specifying coherence protocols
- Challenging to get right, bugs in existing protocols
- Model Checking commonly used for these protocols

---

**Errata**

**AI39.** Cache Data Access Request from One Core Hitting a Modified Line in the L1 Data Cache of the Other Core May Cause Unpredictable System Behavior

**Problem:** When request for data from Core 1 results in a L1 cache miss, the request is sent to the L2 cache. If this request hits a modified line in the L1 data cache of Core 2, certain internal conditions may cause incorrect data to be returned to the Core 1.

**Implication:** This erratum may cause unpredictable system behavior.

**Workaround:** It is possible for the BIOS to contain a workaround for this erratum.

**Status:** For the steppings affected, see the Summary Tables of Changes.
What we measured

- Expression Inference:
  - Does pruning based on concrete snippets help?
- Usability of the specification approach:
  - Tradeoff in using purely concrete vs. symbolic snippets
  - Human effort involved
The protocols we examined occasionally required update expressions of size 15 or more. Pruning enables the algorithm to scale well enough to synthesize these protocols.
Only Concrete Snippets?

- Specified two textbook protocols using only concrete snippets
- On average < 2 snippets required per transition
- Converged on a correct protocol in a few iterations
- Synthesis time: < 1 second
Usability: Case Studies

- Textbook protocol using mixed snippets
- Initial symbolic snippets from informal desc.
- Additional concrete snippets added to fix errors
- Correct protocol in a few iterations

<table>
<thead>
<tr>
<th># of initial snippets</th>
<th>96</th>
</tr>
</thead>
<tbody>
<tr>
<td># of snippets in final version</td>
<td>108</td>
</tr>
<tr>
<td>Total Manual Effort</td>
<td>13 hours</td>
</tr>
<tr>
<td># of debug iterations</td>
<td>8</td>
</tr>
<tr>
<td># of counter-examples inspected</td>
<td>6</td>
</tr>
<tr>
<td># of updates/guards inferred</td>
<td>175/80</td>
</tr>
<tr>
<td># of states in verified protocol</td>
<td>1.5 M</td>
</tr>
</tbody>
</table>
Usability: SGI Origin

• Specified the protocol used in SGI Origin
  • Complex, industrial protocol
• Intermixed concrete and symbolic snippets
• Successfully converged to a correct protocol
• Computational effort:
  • Final synthesis took 30 minutes of CPU time
Conclusions

• **TRANSIT** for specifying distributed protocols
• Straightforward to go from informal specs to snippets
• Allows intermixing of concrete and symbolic snippets
• Uses an explicit enumerative CEGIS approach for synthesis
  • Scalable enough to be useful
• Evaluated usability of specification methodology
  • Used **TRANSIT** to specify the industrial protocols
Thank You

Questions?